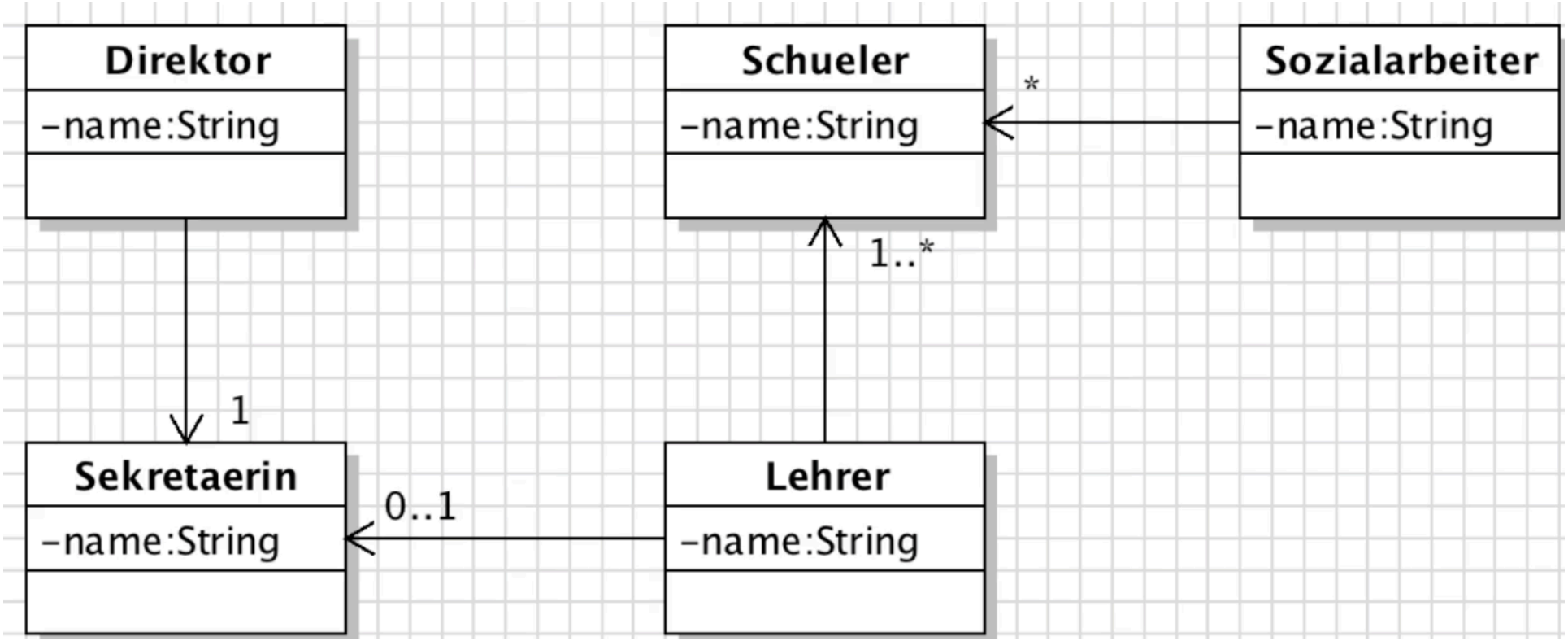
Java:

Implementierung von Assoziationen

Beispielsoftware: "Schulverwaltung"

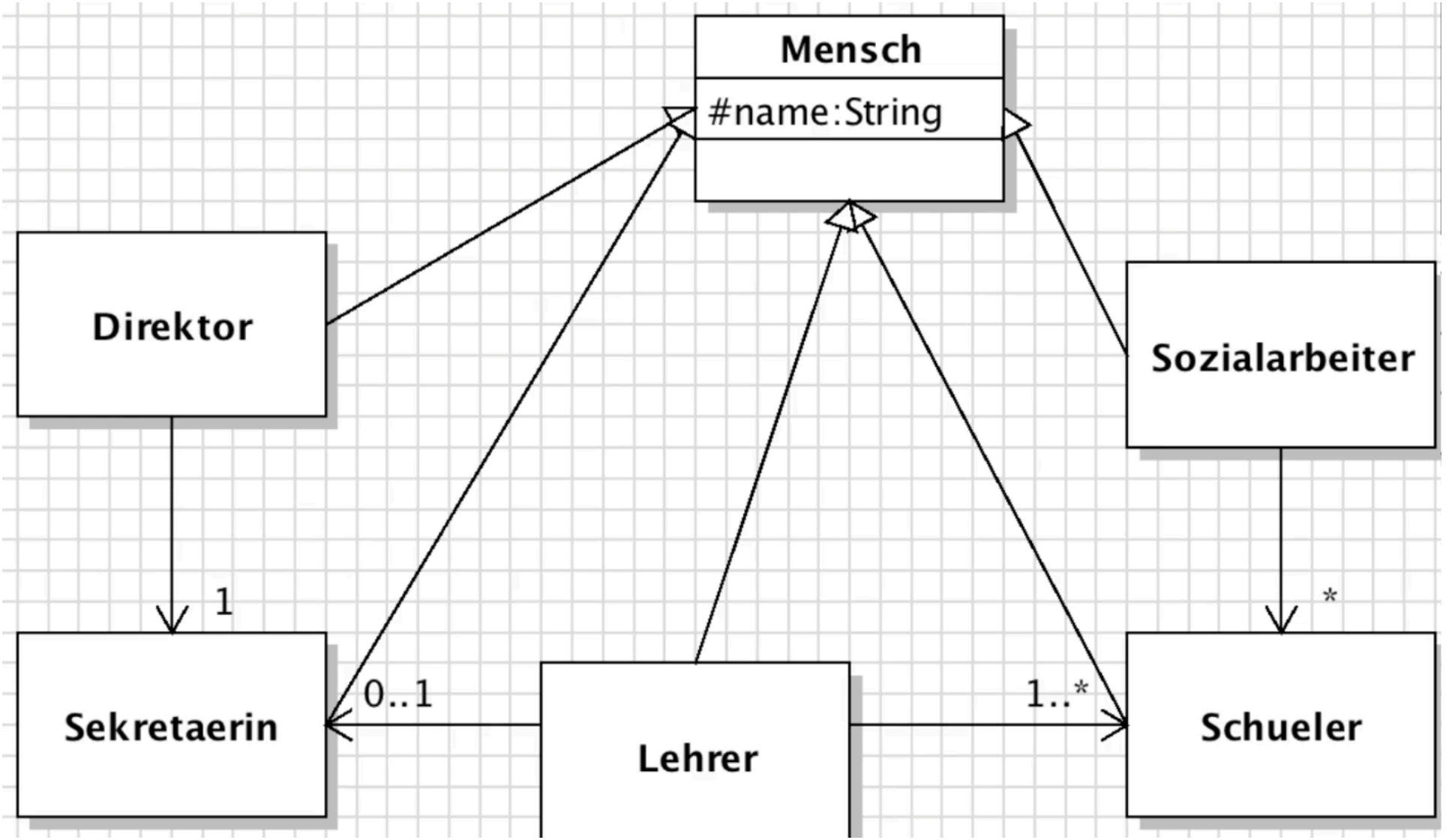
Wie könnten Sie hier Vererbung sinnvoll einsetzen?



Beispielsoftware: "Schulverwaltung"

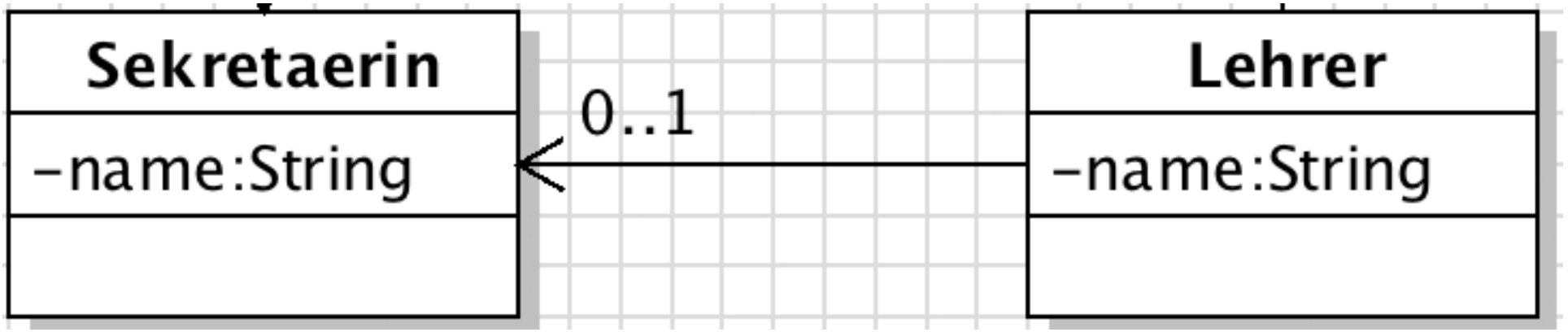
Attribut name:String in Oberklasse auslagern.

(Aber: Das tun wir jetzt nicht, um möglichst einfach zu bleiben)



0...1 - Beziehung

Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Lehrer eine Sekretärin hat (oder keine)?

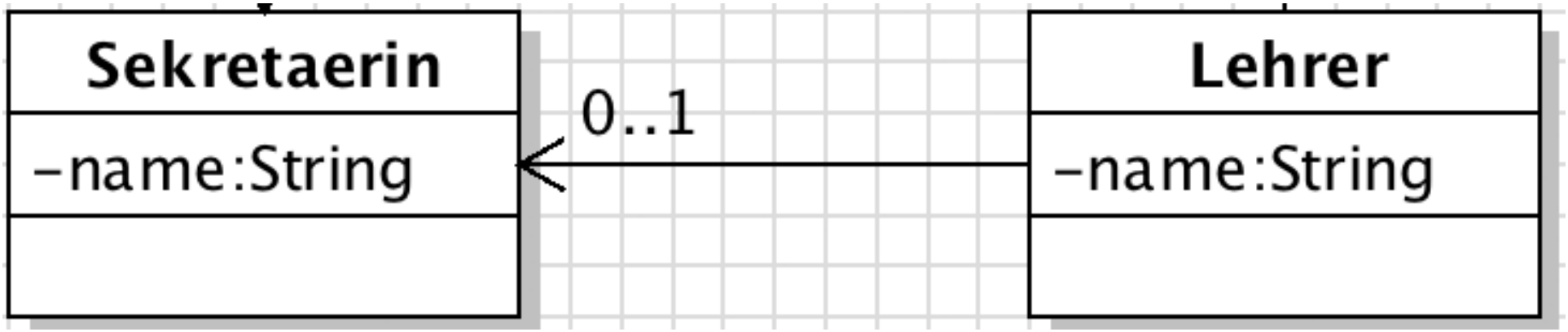


0...1 - Beziehung

Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Lehrer eine Sekretärin hat (oder keine)?

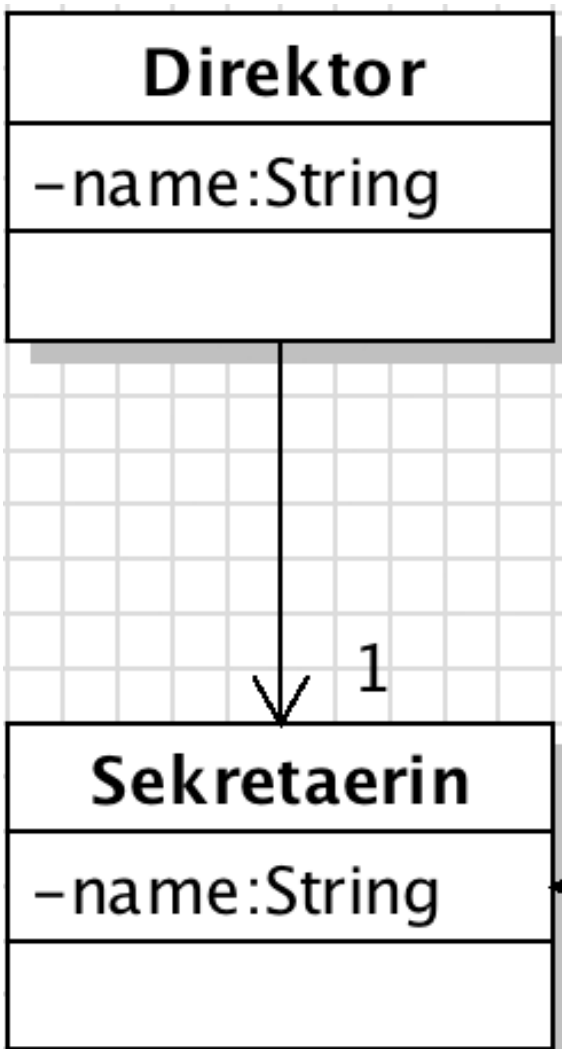
Lösung:
Ein Objekt der verbundenen Klasse als Attribut deklarieren.

```
// Klasse Lehrer
private Sekretaerin sekretaerin;
```



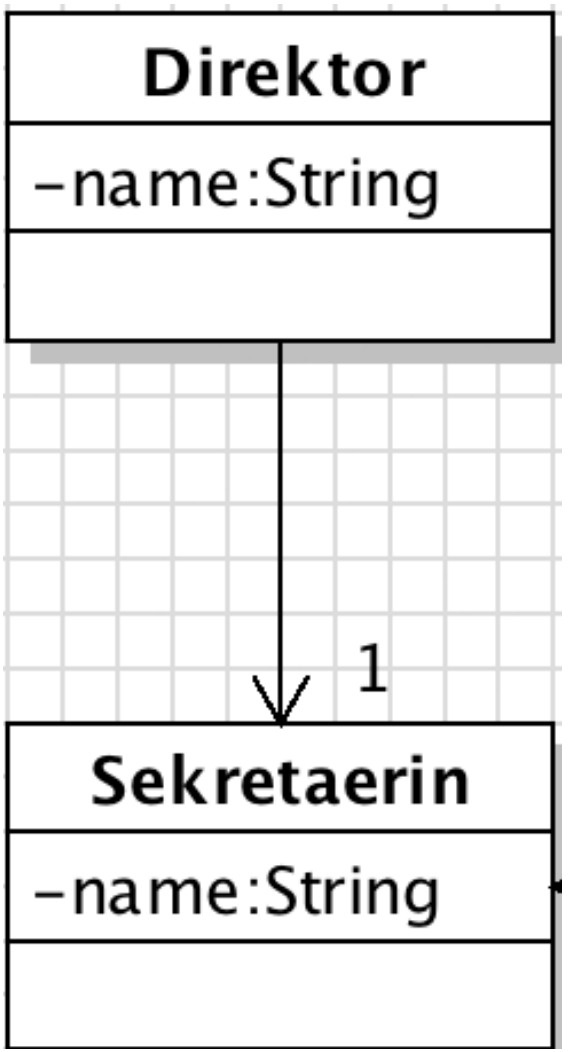
1 - Beziehung

Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Direktor genau eine Sekretärin hat?



1 - Beziehung

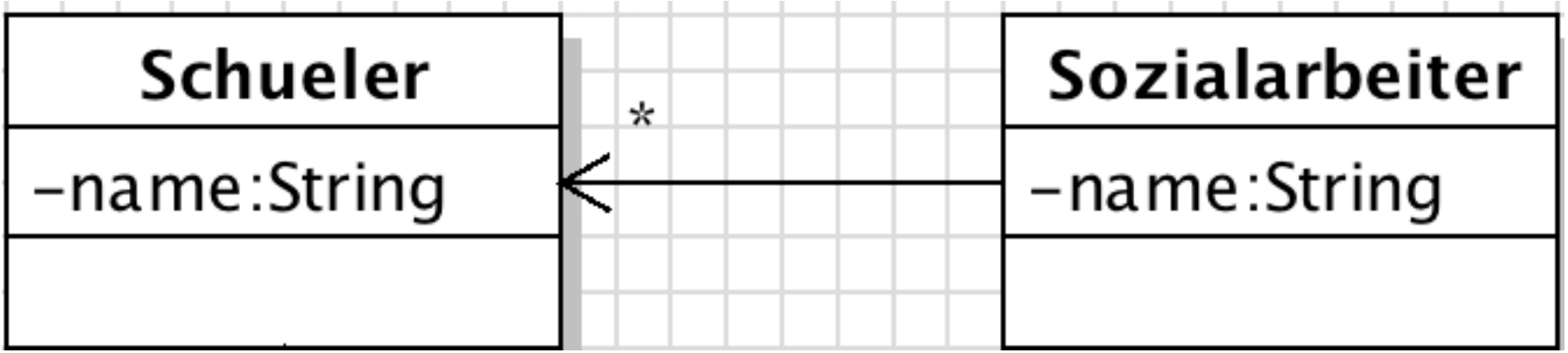
Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Direktor genau eine Sekretärin hat?



Lösung:
Ein Objekt der verbundenen Klasse als Attribut deklarieren und dieses Attribut via parametrisiertem Konstruktor initialisieren.

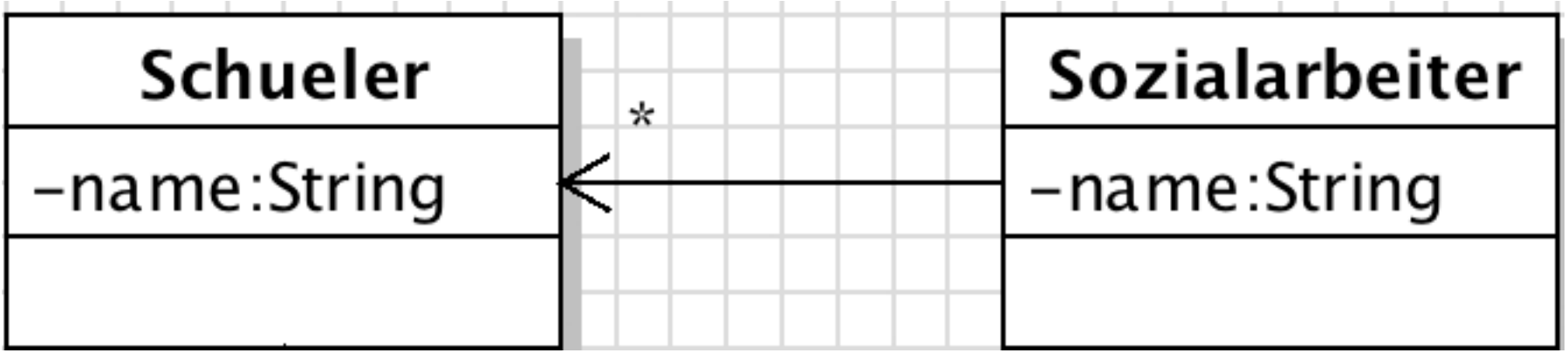
```
// Klasse Direktor
private Sekretaerin sekretaerin;

public Direktor(Sekretaerin s)
{
   this.sekretaerin = s;
}
```

* - Beziehung



Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Sozialarbeiter keinen, einen oder mehrere Schueler betreut ("kennt")?

* - Beziehung



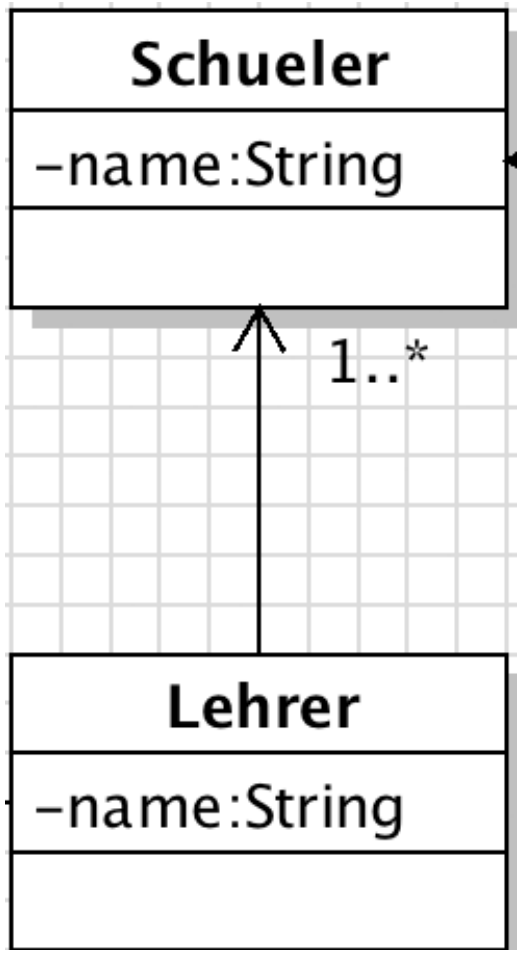
Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Sozialarbeiter keinen, einen oder mehrere Schueler betreut ("kennt")?

Lösung:
ArrayList mit Objekten der verbundenen Klasse anlegen.

```
// Klasse Sozialarbeiter
private ArrayList<Schueler> schuelerliste
        = new ArrayList<Schueler>();
```

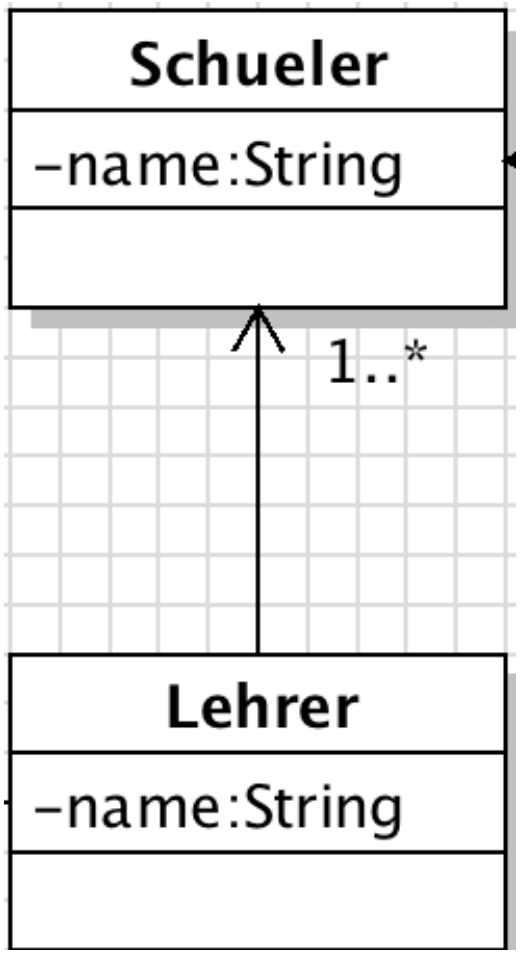
1...* - Beziehung

Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Lehrer mindestens einen Schueler hat?



1...* - Beziehung

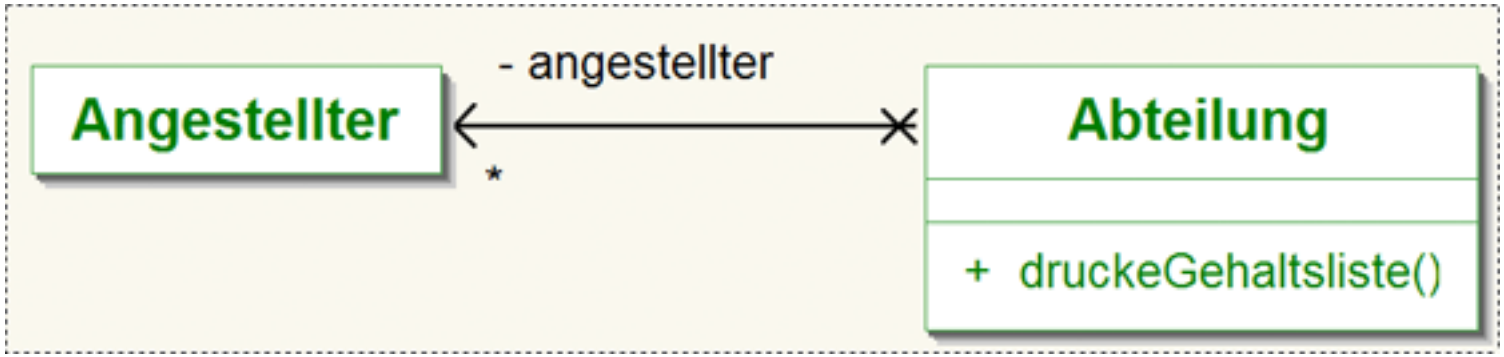
Überlegen Sie:
Wie legen Sie im Code fest, dass jeder Lehrer mindestens einen Schueler hat?



Lösung:
ArrayList mit Objekten der verbundenen Klasse anlegen; Liste im Konstruktor übergeben.

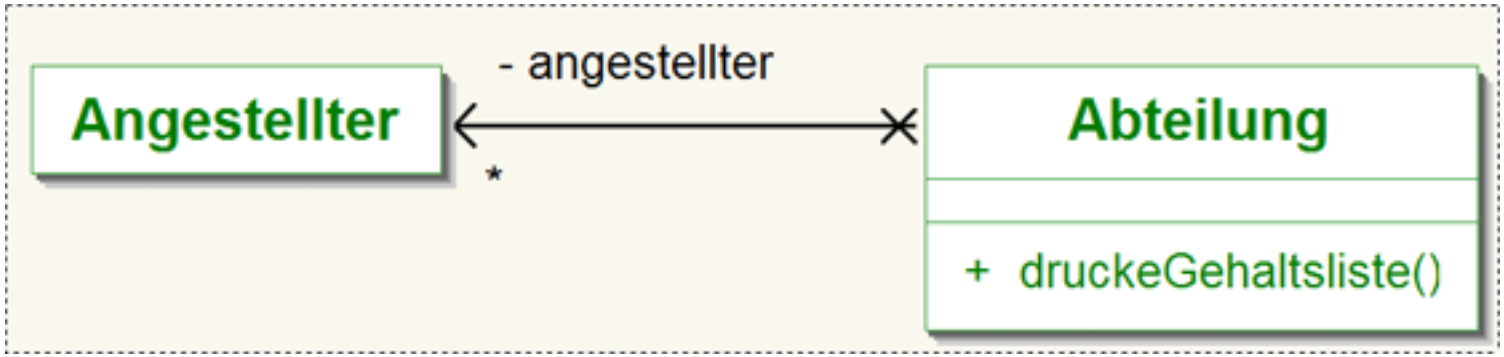
```
// Klasse Lehrer
private ArrayList<Schueler> schuelerliste;

public Lehrer(ArrayList<Schueler> sListe){
   this.schuelerliste = sListe;
}
```

Nicht navigierbar



Überlegen Sie:
Wie legen Sie im Code fest, dass der Angestellte die Abteilung nicht kennt?

Nicht navigierbar



Überlegen Sie:
Wie legen Sie im Code fest, dass der Angestellte die Abteilung nicht kennt?

Lösung:
Sie tun einfach nichts. Wenn Angestellter keine Referenz auf Abteilung enthält, gibt es in diese Richtung keine Sichtbarkeit.

Diverses zur Implementierung

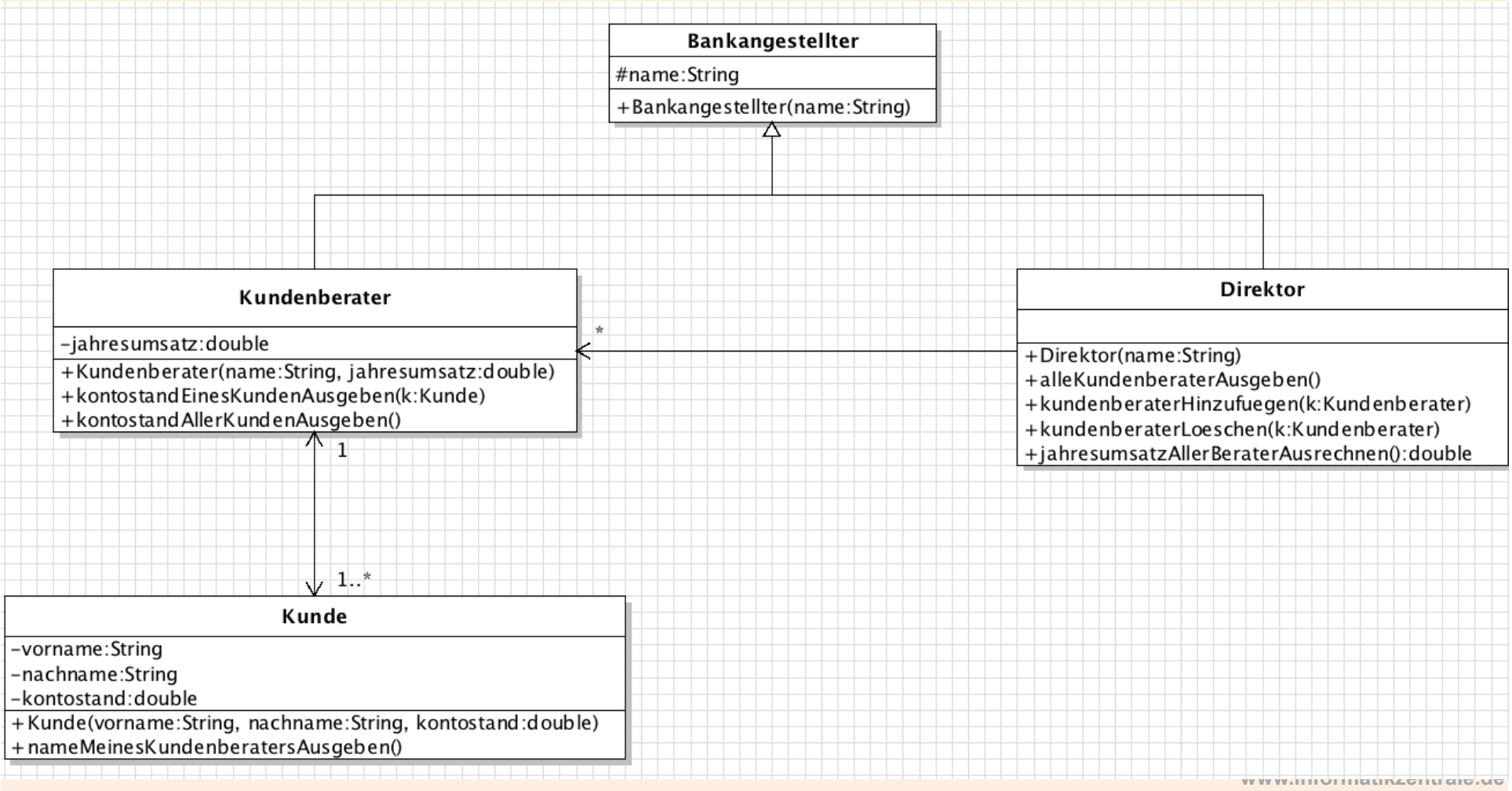
1) Bei bidirektionalen Beziehungen muss berücksichtigt werden, dass bei Änderungs- und Löschvorgängen auch in der assoziierten Klasse entsprechend geändert wird.

2) Änderungs- und Löschoperationen werden i.d.R. durch eigene Methoden vorgenommen.

3) eUML2 erzeugt Ihnen Javacode passend zum UML-Diagramm (Forward-Engineering, vgl. MySQL-Workbench). In der Regel ist der Code brauchbar oder kann zumindest als Grundlage dienen.

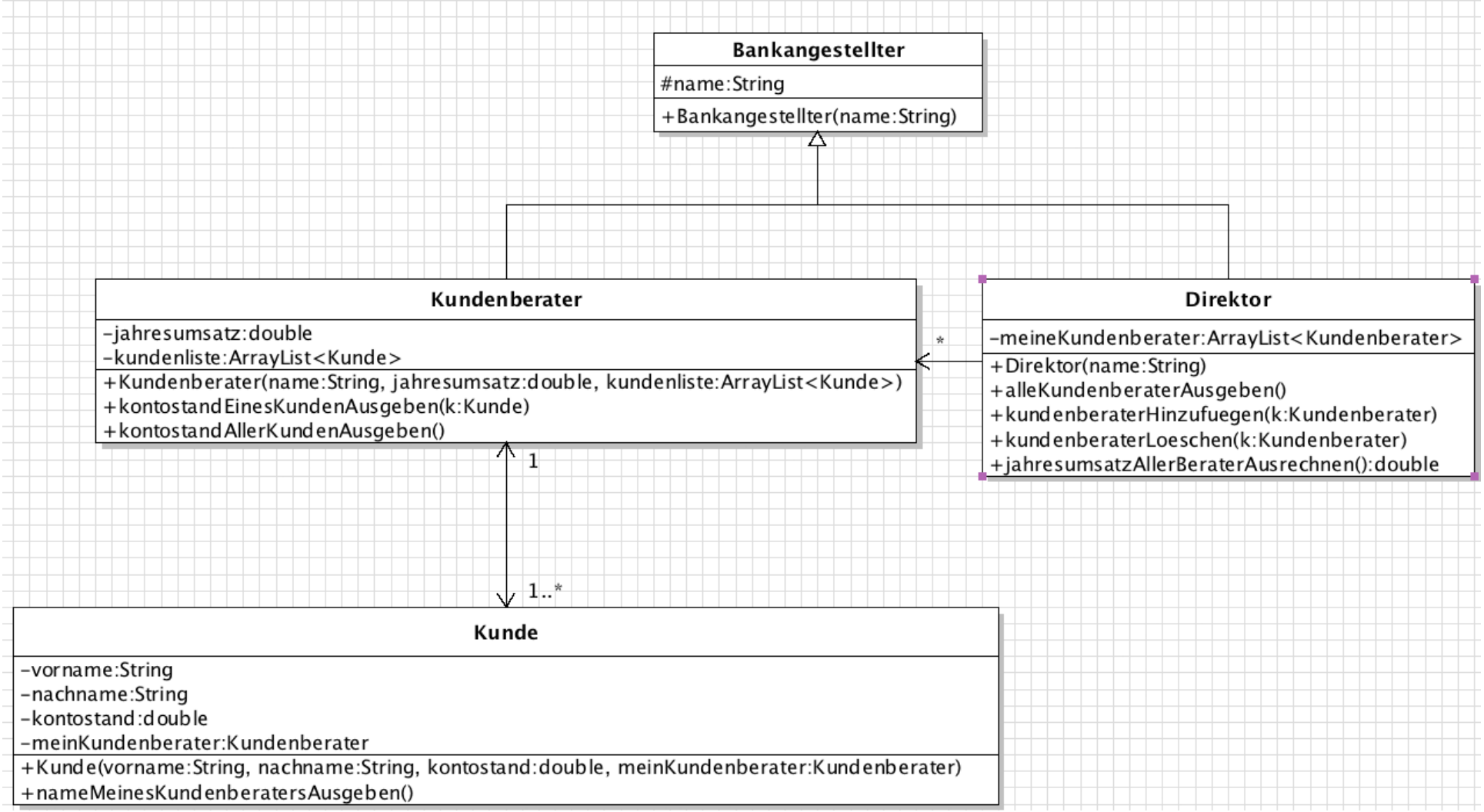
Assoziationen – Übung: Anwendung

Aufgabe 1: Die Assoziationsattribute sind noch nicht festgehalten. Ergänzen Sie das Klassendiagramm entsprechend (auch Parameter in den Konstruktoren bei Bedarf!).



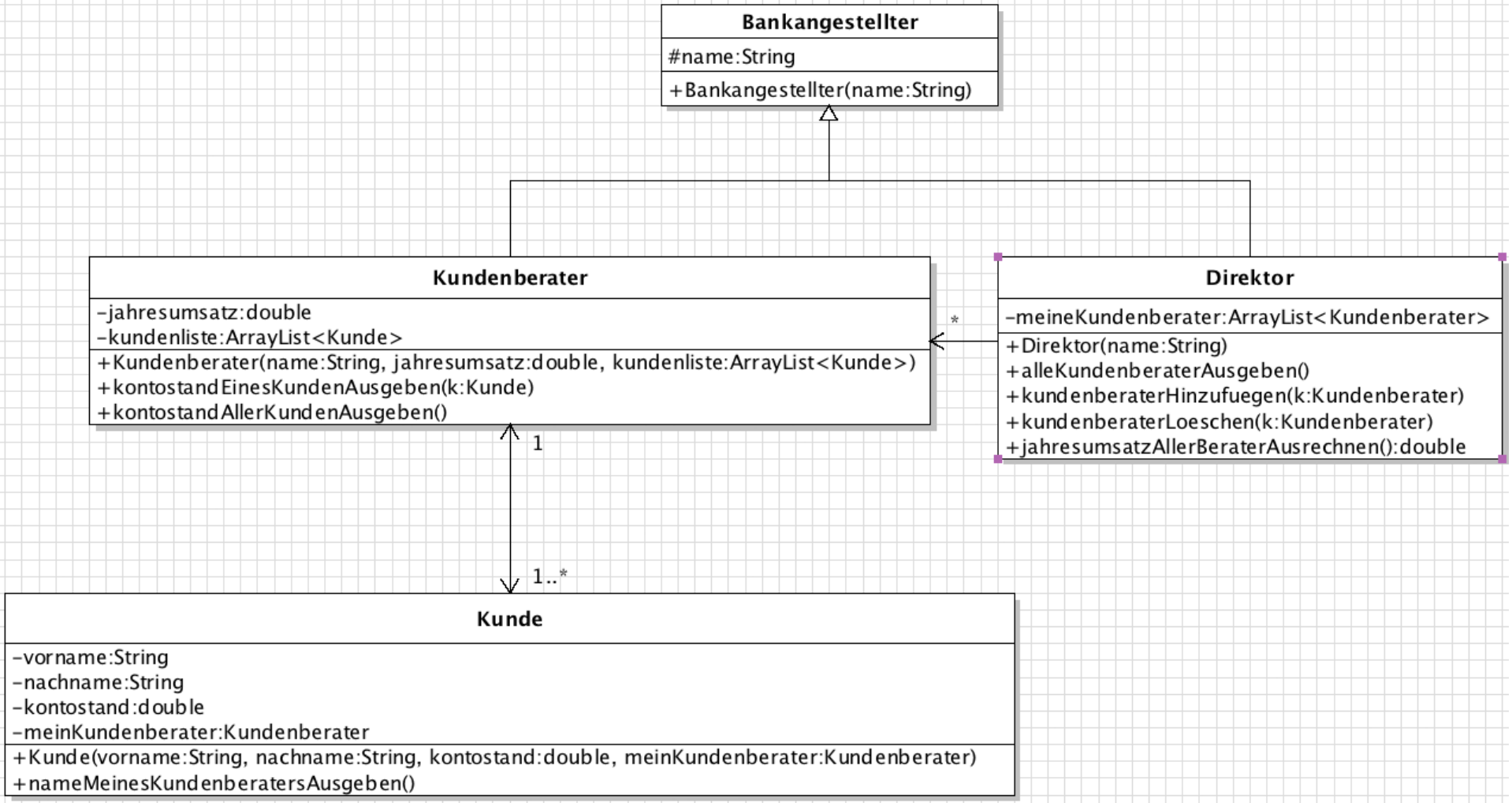
Lösung: # Assoziationen – Übung: Anwendung

- Direktor hat Liste mit Kundenberatern; da Liste leer sein darf (KANN-Beziehung), wird sie nicht im Konstruktor übergeben.
- Kundenberater hat Liste von Kunden; da MUSS-Beziehung, wird sie im Konstruktor übergeben.
- Kunde hat einen Kundenberater (Attribut meinKundenberater). Da MUSS-Beziehung, wird der Kundenberater im Konstruktor übergeben.



Assoziationen – Übung: Anwendung

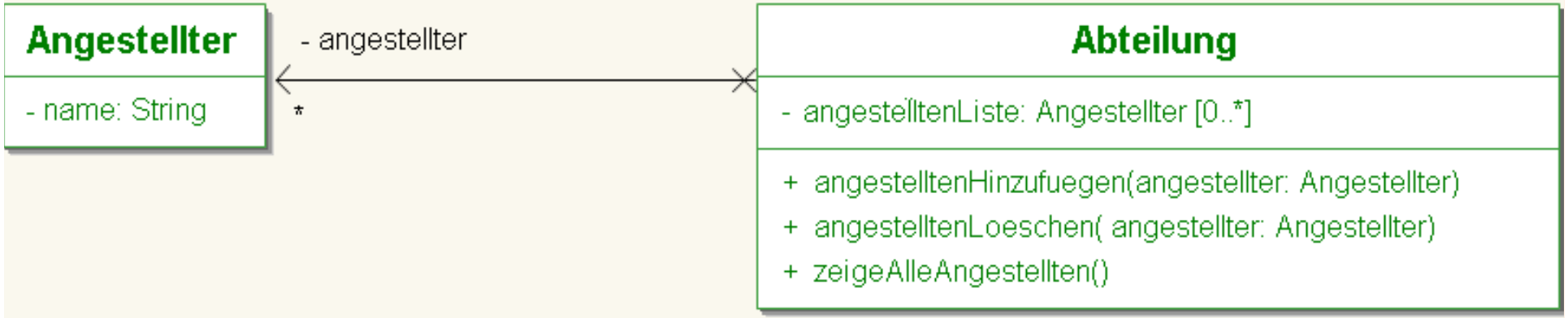
Aufgabe 2: Programmieren Sie die Klassen und testen Sie in der Startklasse ausgiebig alle Methoden, Attribute etc.



Assoziationen – Änderungsmethoden: unidirektional

Setzen Sie dieses Klassendiagramm programmiertechnisch um.
(Natürlich hat jede Klasse zwei sinnvolle Attribute, z.B. "name" o.ä.)

Schreiben Sie außerdem in der Klasse Abteilung Methoden, um neue Angestellte hinzuzufügen oder zu löschen.



Assoziationen – Änderungsmethoden: bidirektional

Setzen Sie dieses Klassendiagramm programmiertechnisch um.
(Natürlich hat jede Klasse zwei sinnvolle Attribute, z.B. "name" o.ä.)

Schreiben Sie außerdem in der Klasse Abteilung Methoden, um neue Angestellte hinzuzufügen oder zu löschen.

Verwenden Sie eclipse!